Evasion

Dennis Shasha

Omniheurist Course

Computer Science

Description

H (the Hunter) wants to catch P (the prey). P wants to evade H as long as possible. You will play both roles (i.e. in the competition, once you will be the hunter and once the prey).

The game is played on a 300 by 300 square. Both P and H bounce off the sides and bounce off any walls created by H. Bouncing is explained below.

P is initially at point (230, 200) and H at position (0,0). In a time step, H moves one unit (always diagonally or counter-diagonally based on its bounce history -- the hunter changes direction only when it encounters a wall). P moves only every other time step. P moves one unit in its current direction unless it hits a wall or it decides to move one unit in some other direction or it chooses not to move. A move of P can be in any direction -- vertical, horizontal, diagonal or counter-diagonal. When both P and H move, they move simultaneously. So, it goes like this:
Timestep 1: Hunter moves
Timestep 2: Hunter and Prey move
Timestep 3: Hunter moves
Timestep 4: Hunter and Prey move

H may create a wall not more frequently than every N time steps (where N is a parameter). When H creates a wall, the wall must touch the point where H was before H moved. The wall must be vertical or horizontal and must touch neither H nor P nor go through another wall (though it may touch another wall). That is, a wall that would hit P if built is not built. After creating a wall, H must wait at least N time steps before attempting to create another wall. It may be of any length. If H tries to create a wall that violates any of these rules, the system will generate a message on the screen, will not build the wall, but will not otherwise penalize H.

H may destroy any wall regardless of where H is. At any given time the maximum number of walls there can be is M. (N and M will be specified the time of the competition.) H may destroy walls at will -- without waiting. In every time step, all creation or destruction of walls occurs before any player moves in that time step.

H catches P if H is within four units of P (based on Euclidean distance) and there is no wall between them. H's score is the number of time units it takes to catch P. Less is better.

Technical note about bouncing: The walls are one unit thick, and centered at integer coordinate. For any integer coordinate that is within a wall, the wall extends (at least) .5 units above, below, left, and right of it. So, if there is a wall from (10,200) to (300,200) and a player begins at (100, 199) moving +1 in x and +1 in y, it will reflect to (101, 199) at the end of the time step. Neither player will bounce off the narrow side of a wall (except the special case described below). So, a player who begins at (9, 200) moving +1 in x and +1 in y, it will not reflect and will simply move to (10, 201), still moving +1 in x and +1 in y. A player who hits a corner of a wall will bounce off as if they hit the middle of the wall. So, a player who begins at (9, 199) moving +1 in x and +1 in y will reflect and end up at (10, 199) moving +1 in x and -1 in y.

Note the following cases:

- The hunter builds a wall, let's say horizontally and to the left and right of him, but also bounces off a horizontal wall. In this case, the wall is simply not built because it would "squish" the hunter between walls.
- The prey tries to move directly into the narrow side of the wall. For example, if the prey tries to move left into a horizontal wall, it will bounce bak to its original position.
- There are two parallel walls, again let's say horizontally, with one centered 1 unit above the other and both ending, let's say on the right, at the same x-coordinate. For example, if one horizontal wall ends at (40, 40) and another horizontal wall ends at (40, 41). Then if the Hunter is at (41, 41) moving -1 in x and -1 in y, it will bounce off as if it hit a vertical wall, and end up at (41, 40) moving +1 in x and -1 in y.

Super-technical: An algorithm for bouncing

```
Look at the square you would be moving to.
If there is no wall there:
        Simply move there and you are done
Else If there is a horizontal wall there:
        Don't move vertically at all
Else if there is a vertical wall there:
        Don't move horizontally at all
If there was a wall there:
        Look at the new square you would be moving to.
        If there is no wall there:
                Simply move there and you are done
        Else If there is a wall there:
                Change which direction you were still moving it
               (i.e. if you had tried bouncing off a horizontal wall before,
                try bouncing off it as if it was a vertical wall now)
                (look at Note 3 above)
                Look at the new new square you would be moving to.
                        If there is a wall there:
                                You don't move at all and you are done
                                (since it was a corner)
                        Else if there is no wall there:
                                Simply move there and you are done
```

Architect

The architect will provide a graphical display showing the walls and the movements of H and P. The game should be designed to allow programs to play together (for the competition) or program against human or human against human (for the final project). The architect reports the positions of H to P and to P of H and also the position of all walls. The architect also records the score and determines if it is impossible for H to find P. The graphics should show the positions of H and P, the walls, and the current time as measured in number of steps taken. You might find the following code helpful.

The score is the number of steps it takes for the hunter to catch the prey. Team A beats team B, if it takes fewer steps for the hunter of team A to catch the prey of team B when A is playing the role of the hunter than it does for the reverse situation.